# CD LAB 5

NAME: Janvii RV

SRN: PES2UG22CS232

DATE: 20/03/2025

Code:

lexer.l :

%{

#define YYSTYPE char\*

#include <unistd.h>

#include "parser.tab.h"

#include <stdio.h>

extern void yyerror(const char \*); // declare the error handling function

%}

/\* Regular definitions \*/

digit [0-9]

letter [a-zA-Z]

id {letter}({letter}|{digit})\*

digits {digit}+

opFraction (\.{digits})?

opExponent ([Ee][+-]?{digits})?

number {digits}{opFraction}{opExponent}

%option yylineno

%%

\/\/(.\*) ; // ignore comments

[\t\n] ; // ignore whitespaces

"(" {return \*yytext;}

")" {return \*yytext;}

"." {return \*yytext;}

"," {return \*yytext;}

"\*" {return \*yytext;}

"+" {return \*yytext;}

";" {return \*yytext;}

"-" {return \*yytext;}

"/" {return \*yytext;}

"=" {return \*yytext;}

">" {return \*yytext;}

"<" {return \*yytext;}

{number} {

yylval = strdup(yytext); //stores the value of the number to be used later for symbol table insertion

return T\_NUM;

}

{id} {

yylval = strdup(yytext); //stores the identifier to be used later for symbol table insertion

return T\_ID;

}

. {} // anything else => ignore

%%

parser.y

%{

#include "quad\_generation.c"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define YYSTYPE char\*

void yyerror(char\* s); // error handling function

int yylex(); // declare the function performing lexical analysis

extern int yylineno; // track the line number

int yywrap() {

return 1; // Signals end of input

}

FILE\* icg\_quad\_file;

int temp\_no = 1;

%}

%token T\_ID T\_NUM

/\* specify start symbol \*/

%start START

%%

START : ASSGN {

printf("Valid syntax\n");

YYACCEPT; // If program fits the grammar, syntax is valid

}

/\* Grammar for assignment \*/

ASSGN : T\_ID '=' E { //call quad\_code\_gen with appropriate parameters

quad\_code\_gen($1, $3, "=", "");

}

;

/\* Expression Grammar \*/

E : E '+' T { //create a new temporary and call quad\_code\_gen with appropriate parameters

$$= new\_temp();

char\* op =strdup("+");

quad\_code\_gen($$,$1,op,$3);

}

| E '-' T { //create a new temporary and call quad\_code\_gen with appropriate parameters

$$= new\_temp();

char\* op =strdup("-");

quad\_code\_gen($$,$1,op,$3);

}

| T

;

T : T '\*' F { //create a new temporary and call quad\_code\_gen with appropriate parameters

$$= new\_temp();

char\* op =strdup("\*");

quad\_code\_gen($$,$1,op,$3);

}

| T '/' F { //create a new temporary and call quad\_code\_gen with appropriate parameters

$$= new\_temp();

char\* op =strdup("/");

quad\_code\_gen($$,$1,op,$3);

}

| F

;

F : '(' E ')' { $$= strdup($2); }

| T\_ID { $$= strdup($1); }

| T\_NUM { $$= strdup($1); }

;

%%

/\* error handling function \*/

void yyerror(char\* s)

{

printf("Error :%s at %d \n",s,yylineno);

}

/\* main function - calls the yyparse() function which will in turn drive yylex() as well \*/

int main(int argc, char\* argv[])

{

icg\_quad\_file = fopen("icg\_quad.txt","w");

yyparse();

fclose(icg\_quad\_file);

return 0;

}

quad\_generation.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "quad\_generation.h"

void quad\_code\_gen(char\* a, char\* b, char\* op, char\* c)

{

    //use fprintf to output the quadruple code to icg\_quad\_file

    printf("%s, %s, %s, %s\n", op, b, c, a);

    fprintf(icg\_quad\_file, "%s %s %s %s\n", op, b, c, a);

}

char\* new\_temp()    //returns a pointer to a new temporary

{

    char\* temp = (char\*)malloc(sizeof(char)\*4);

    sprintf(temp, "t%d", temp\_no);

    ++temp\_no;

    return temp;

}

quad\_generation.h

extern FILE\* icg\_quad\_file;     //pointer to the output file

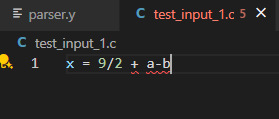
extern int temp\_no;             //variable to keep track of current temporary count

void quad\_code\_gen(char\* a, char\* b, char\* op, char\* c);

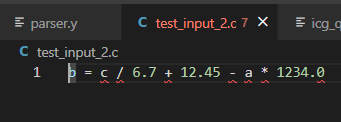
char\* new\_temp();

Input:

1:

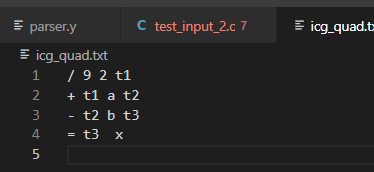


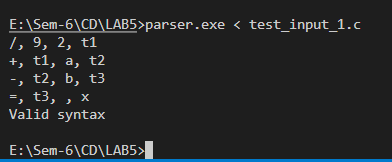
2:



Output Screenshot:

1:





2:

